API document of USB Camera for Unity Android (v1.4)

Android support

Coexist with **Vuforia** and **Oculus Quest**

Preview support in the **Editor** and **Win Standalone**

System Requirements:

Build Android: API level 21 or later with OTG function (test on 6.0, 8.0 and 9.0)

Support limited preview in Editor and Win Standalone

USB Camera for Unity is an Assets Plugin for using USB camera with Unity Android.

Features:

Support multiple USB cameras: 1 to 4 USB cameras

Support dynamic resolution, parameters (brightness etc.) adjustment and hot plugging

Support video streaming with USB and native camera for all platforms except WebGL

Support photo capture and album with multiple USB cameras

Email: chaosikaros@outlook.com

1.1: AARplugin:

Class name: AARplugin	
Major functionality	provide basic function of USB Camera
Important attributes	public const int FRAME_FORMAT_DEFAULT = 1: MJPEG format public const int FRAME_FORMAT_YUYV = 0: YUYV format public const int FRAME_FORMAT_MJPEG = 1: MJPEG format public USBCamera[] cameraScreens = new USBCamera[4]: camera object public Texture2D[] rawTextures = new Texture2D[4]: raw texture of camera frame public bool PromiscuousModeOn = false: whether to enable Promiscuous Mode
Important methods	public void InitPlugin(): call this function to initialize camera format and USBCamera Manager. public bool SetPromiscuousMode(bool input): In Promiscuous mode, the plugin will recognize all USB devices as a USB camera. Do not enable this mode unless you failed to connect to your camera. public bool CreateUSBCameraFormat(int deviceID, int i_frameFormat = FRAME_FORMAT_YUYV, int i_width = 640, int i_height = 480, int i_minFPS = 1, int i_maxFPS = 30, float i_bandwidthFactor = 1.0f): call this function to create USB Camera Format with specific device ID (0-3). i_bandwidthFactor: use to evaluate the final bandwidth of the USB camera. public bool CreateUSBCameraManagerWithSpecificDevices(string deviceName1, string deviceName2 = "usb/001/004", string deviceName3 = "usb/003/005", string deviceName4 = "usb/003/006"): call this function to create USB Camera Manager with specific device names. Devicename: you can get the device name from the Android Toast after you connect the USB camera. public bool CreateUSBCameraManagerWithRandomDevices(string deviceName1 = "usb/001/002", string deviceName2 = "usb/001/004", string deviceName3 = "usb/003/005", string deviceName4 = "usb/003/006"): call this function to create USB Camera Manager with random devices. public Texture2D GetFrame(int deviceID): call this function to get the Texture2D of camera frame with specific device ID. public void RefreshCameraStates(int deviceID): call this function to refresh camera states with specific device ID.

1.2: USBCamera:

Class name: USBCamera	
Major functionality	provide interface of USB camera plugin
Important attributes	public Album album: album object public bool playing = false: camera state public bool stopping = false: camera state public AARplugin plugin: plugin object public int deviceID = 0: device ID public int width = 640: width of camera public int height = 480: height of camera public int FPS = 30: FPS of camera public GameObject screen: use 3D model as a camera screen public GameObject screenUI: use 2D UI as a camera screen public RawImage screenImage: image container of 2D camera screen public MeshRenderer screenRender: image container of 3D camera screen public GameObject detailsPanel: UI panel of parameters public Slider brightnessSlider: slider of brightness public Slider contrastSlider: slider of contrast public bool onlyRenderOnUI = false: whether to only render camera image on UI public bool isUSBCamera = false: whether to connect USB camera public bool autoFit = true: whether to enable auto fit of camera screen public int frameID = 0: frame counter private WebCamTexture webCamTexture: use to preview on Editor or Windows public List<string> supportedSizes = new List<string>(): List of supported resolutions of USB camera public Dropdown sizeSelector: dropdown of resolutions public string currentSize = "": current resolution public Texture2D tempTexture2D: current Texture2D of camera frame

	public Text FPS_text: text display of FPS
Important methods	public void InitScreen(): call this function to initialize camera screen public void InitSupportedSizes(string rawConfigs): call this function to initialize supported resolutions of USB camera. public void GetSupportedSize(): call this function to get supported resolutions. public bool RequireSupportedSize(): call this function to require supported resolutions. public void OnSizeChanged(int value): call this function to change resolution of USB camera. public IEnumerator InitCamera(): call this function to initialize USB camera for Editor or Windows public IEnumerator InitCameraForAndroid(): call this function to initialize USB camera for Android. public IEnumerator FPSCounter(): call this function to calculate FPS. public void AutoFitScreen(GameObject screenObject, float weight): call this function to implement auto fit screen. public void SwitchDetailsPanel(): call this function to switch panel of parameters. public void OnBrightnessSliderChange(float value): call this function when the brightness change. public void OnContrastSliderChange(float value): call this function when the contrast change. public void TakePhoto(): call this function to take a photo. public Texture2D TextureToTexture2D(Texture texture, bool setWithCameraSize = false): call this function to convert texture to texture2d.

1.3: Album:

Class name: Album	
Major functionality	provide interface of album
Important attributes	public int previewWidth = 640: default preview width in album public int previewHeight = 480: default preview width in album public GameObject albumRoot: root of album public GameObject photoPrefab: prefab of photos public GameObject albumParent: parent of albums public Scrollbar albumScrollbar: scrollbar of album scrollbar public GameObject previewPanel: panel of photo preview public RawImage previewImage: image for preview photo public Text pathText: show the file path of preview photo public static Photo currentPhoto: current preview photo public static bool startPreview = false: state of album private string savePath: save path of all photos public List<string> fileList = new List<string>(): file list of all photos private List<Photo> photoContainer = new List<Photo>(): container list of all photos
Important methods	public void InitAlbum(): call this function to initialize album. public void SwitchAlbum(): call this function to switch album panel. public void LoadPhotos(string parentPath): call this function to load all photos. public void SavePhoto(Texture2D rawPhoto): call this function to save a photo from a texture2D. public void DeletePhoto(): call this function to delete current preview photo. public void SwitchPreviewPanel(): call this function to switch preview panel.

1.4: Photo:

Class name: Photo	
Major functionality	provide interface of Photo
Important attributes	public RawImage screenImage: render image of this photo. public string filename: file path of this photo.
Important methods	public void OnPreview(): call this function to preview this photo.

1.5: StreamingServer:

Class name: StreamingServer	
Major functionality	provide function of video streaming server
Important attributes	public enum CompressMode: compress mode for video compressing public enum StreamingState: state of streaming public USBCamera usbCamera: usb camera public CompressMode compressMode: current compress mode public StreamingState streamingState: current streaming state public RawImage screen: video source public string IP: IP address of server public int port: the port of server is same with the one of client public int senderFPS: refresh rate of server public TcpListener serverListener: tcp listener of server public Dropdown IPSelector: dropdown for IP selection public Text FPS_text: debug text of frame information public bool enableFPSDisplay: whether to display FPS information public bool stop: whether to stop the server public bool compressdFormat: whether to use a compressed format public List<string> IPList: list of all available IP addresses public int videoQuality: video quality of sending frame (1 - 100) public int compressFormat = 0: current format of compressed texture

	public int frameID: ID counter of frame public int broadcastingRate: broadcasting rate (ms) of server private Texture2D sentTexture2D: texture2D to be sent private RenderTexture currentRT: cache of current render texture private RenderTexture renderTexture: current render texture private byte[] frameMsg: protocol of each frame private byte[] rawBytes: raw byte array of texture from each frame private Thread connectThread: loop thread of server public static int frameMsgLength: length of frame protocol
Important methods	public void SetVideoQuality(float i): call this function to adjust video quality. public void InitServer(Text text = null): call this function to initialize server. public async void ConnectLoop(): call this function to launch the loop of server. public void RefreshIP(): call this function to refresh IP selector and IP list. private async Task Accept(TcpClient client): call this function to process each connection from client. public void OnIPChanged(int value): listener of IP selector. public IEnumerator ServerLoop(): coroutine of server loop. public IEnumerator FPSCounter(): coroutine of FPS counter. public void RestartServerLoop(): call this function to restart the server loop. public void Feedback(NetworkStream networkStream): call this function to receive feedback from client. public void SendFrameMsg(NetworkStream networkStream): call this function to send protocol of each frame. public void SendFrameData(NetworkStream networkStream): call this function to send frame data of each frame. public void RefreshFrameData(): call this function to refresh frame data. public Texture2D TextureToTexture2D: call this function to control the quality of each frame.

1.6: StreamingClient:

Class name: StreamingClient	
Major functionality	provide function of video streaming client
Important attributes	public enum DecompressMode: decompress mode for video decoder public enum StreamingState: state of streaming public DecompressMode decompressMode: current decompress mode public static StreamingState streamingState: current streaming state public RawImage screen: screen of received frame public string IP: IP address of server public int port: the port of client is same with the one of server public int senderFPS: refresh rate of client public Text inputText: input text of IP address public Text FPS_text: debug text of frame information public bool decompressFrame: whether to decompress (rescale) each frame public bool enableFPSDisplay: whether to display FPS information public bool stop: whether to stop the client public int frameWidth : width of received frame public int frameHeight: height of received frame public int videoQuality: video quality of received frame (1 - 100) public int compressFormat = 0: current format of compressed texture public int frameID: ID counter of frame public static int frameMsgLength: length of frame protocol private List<Frame> cachedFrames: list of cached frames private Texture2D receivedTexture2D: received texture2D private Texture2D receivedTexture2Dtemp: cache of received texture2D private RenderTexture currentRT: cache of current render texture private RenderTexture renderTexture: current render texture private bool compressedFormat: whether the received frame used

	compressed format private int cachedFrameCounter: counter of cached frames private float scale = 1.0f: scale factor of each frame private TextureFormat currentFormat: current texture format private bool restarting = false: whether the client is restarting
Important methods	public void InitClient(Text text = null): call this function to initialize the client. public void AutoFitScreen(GameObject screenObject, float weight = 1.0f): call this function to apply auto fit on screen objet. public IEnumerator ClientLoop(): coroutine of client loop. public IEnumerator FPSCounter(): coroutine of FPS counter. public void RestartClientLoop(): call this function to restart the client. public void DisplayFrame(Frame frame): call this function to display received frame. public Texture2D DeCompressFormat(Texture2D texture): call this function to decompress a texture with compressed format. public Texture2D DeCompressTexture(Texture2D texture, float scale = 1.0f): call this function to rescale received texture2D. public void ClearCachedFrames(): call this function to clear cached frames.

1.7: Frame:

Class name: Frame	
Major functionality	provide function of frame receiver
Important attributes	public TcpClient receiverClient: tcp client of receiver client public string filename: file path of this photo. public int frameWidth : width of received frame public int frameHeight: height of received frame public int frameLenght: length of received raw frame byte array public int videoQuality: video quality of received frame (1 - 100) public int compressFormat = 0: current format of compressed texture

	public int frameID: ID counter of frame public byte[] rawBytes: raw byte array of texture from each frame public bool emptyFrame: whether the received frame is empty public bool downloaded: whether the download is completed public bool used: whether the frame has been displayed private bool connected: whether the connection is successful. private NetworkStream serverStream: network stream from server private string rawMsg: raw string of frame protocol private byte[] frameMsg: raw byte array of frame protocol
Important methods	public Frame(int frameID): constructor of Frame. public void StartReceive(string IP, int port): call this function to start connection. public void ReceiveFrameMsg(): call this function to receive frame protocol. public void ReceiveFrame(): call this function to receive frame data. public void Feedback(): call this function to give feedback to server. public void Close(): call this function to close connection.

1.8: ThreadManager:

Class name: ThreadManager	
Major functionality	provide function of thread manager
Important attributes	public class DelayedAction: class of a delayed action public static int maxThreads: max threads of thread manager public static int threadCounter: counter of thread public static bool exist: whether a thread manager has been created public static ThreadManager threadManagerHolder: holder of thread manager private List<Action> tempActions: cache of created actions private List<Action> currentActions: cache of tempActions private List<DelayedAction> delayedActions: cache of delayed actions

	private List<DelayedAction> currentDelayedActions: cache of delayedActions public static ThreadManager threadManager: Singleton of thread manager
Important methods	public static void InitThreadManager(): call this function to initialize thread manager public static void RunUnityAction(Action inputAction, float delayedTime = 0): call this function to run an delayed action which contains Unity API in main thread of Unity. public static Thread RunOrginalAction(Action inputAction): call this function to run an action without unity API in other thread. private static void RunAction(object action): call back of thread pool. ThreadManagerLoop(): call this function to run delayed actions with Unity API.

1.9: IPManager:

Class name: IPManager	
Major functionality	provide function of IP provider
Important methods	public static string GetIPList(): call this function to get all IP address within inter network.

2.0: CameraDebug:

Class name: CameraDebug	
Major functionality	provide function of Debug Log control
Important attributes	public static bool enableDebug : whether to enable Debug
Important methods	public static void Log(string info) : call this function to log information under this plugin